4-Other Client Services

Other Client Services

This section describes the configuration properties of other ACE objects under Client Services, other than the Master Channel and Internal Channel. Generally speaking, a "client service" is a network or other process that initiates connections to another server or device.

MQTT Publish (MQClient, MQ_Extra_Clients)

The MQ Publish object defines some of the properties of a service which sends an unsolicited data packet to a Broker using the MQTT protocol. This service is used in conjunction with ISaGRAF program logic to publish data from an RTDB to the broker using TCP/IP.

The RTDB allows for up to four report-by-exception processes, including primary/secondary HCP, and one or more MQ RBE objects. The MQClient provides a child object that can be configured as either a primary or secondary RBE process. The MQ_Extra_Clients child object allows either the tertiary (3rd) or quaternary (4th) RBE processes to be used, for systems which need to report both to an HCP and with MQ RBE.

Attributes	Function
Object Type	RtdbSegmentFdb
Parent(s)	System Clients
Instance	Must be between 0 for MQClient, between 0 and 1 for MQ_Extra_Clients.

Properties	Values
Connection Type	Select the method by which the MQ connection is established to the broker. • Persistent - Connection is made to the broker and is permanently established, with data sent over the established connection as needed. • Non-Persistent - Connection is made to the broker as needed, and disconnected after pending data has been sent.
Retry Period	Time (in milliseconds) to wait for published message to be acknowledged before retrying.
Keep Alive	Time (in milliseconds) between "ping" messages (MQtt-keep alive packet) in order to ensure the integrity of a Persistent connection. For Non-persistent connections, this is the amount of inactivity time before closing the socket to the broker.
Client ID (Login Name)	Client ID to uniquely identify this device to the MQTT Server upon connection. Make sure that all RediGates and all other clients connecting to the server have unique IDs. *Enter an MQTT Client ID between 1 and 23 alphanumeric characters. If the server supports the MQTT protocol version 3.1.1 or higher, the Client ID may be up to 31 characters and may include other characters besides alphanumeric.* *If the Client ID field is set to **${GATEWAY}** the Linux Unit Name from the top-level System Configuration object is used as the Client ID.* *If the Client ID field is left blank, the Client ID will be taken from a random value in the Linux file system (`/proc/sys/kernel/random/uuid`). This will guarantee all gateway devices have a unique Client ID without requiring special configurations, as long as the MQTT broker doesn't have any specific requirements as to the contents of the Client ID.* *If the Client ID field is set to **${MQTT_LOGIN}** and an entry is included in the GlobalText object with something like:* *Search Name=MQTT_LOGIN, Replacement Text=This_Is_A_Long_Client_ID_Up_To_512_Bytes* *then then Client ID field will be used from the GlobalText object entry with het name "MQTT_LOGIN". This is useful if the MQTT system requires a Client ID that is longer than 31 characters.*

Last Will Topic	This is used with the Last Will Payload (below) to inform a subscribed host that the MQTT session is not currently connected and is thus not reporting live data. *T* *he Last Will Topic is the MQ topic sent by the broker as part of a Last Will & Testament message. It typically identifies the unit and identifies with the topic of a Last Will & Testament message such as:* *RBE/ConsoleID/DEATH/unitname_or_number* *If any of the following parameter strings is included in the Last Will Topic, the information from the configuration will be substituted in place of the parameter:* • *${REDIGATE} ${REDIGATE} is substituted with the Unit Name from the top-level System object.* • *${DIRECTOR} ${DIRECTOR} is substituted with the Unit Name from the top-level System object.* • *${GROUP} ${GROUP} is substituted with the Console_ID field of the MQRbePr child object.*
Last Will Payload	The Last Will Payload is the payload of the message sent by the broker to the subscribers whenever the MQTT connection has been lost. It allows the host to know that the data is no longer being updated in real time and should be considered "stale". *The default topic for Last Will Topic is "**RBE/${GROUP}/DEATH/${GATEWAY}**", where "${GROUP}" is substituted from the Console_ID in the MQ RBE object, and the "${GATEWAY}" is substituted from the Unit Name in the System object. This is the standard convention when using the RediGate MQ-RBE protocol, and failing to set the Last Will Topic properly may prevent the MQTT host from marking the RediGate alive upon connection.* *For other MQTT host types other than MQ-RBE (such as JSON), see RediGate Quick Start documents on how the MQTT settings should be configured.*
Connection Delay	If configured with more than one MQTT server address, this is the time (in milliseconds) between a closed or failed broker connection until the next connection attempt.
Connection Port	Port number at the broker to connect for MQTT data transactions. *Typically port 1883 is used for unencrypted connections, or port 8883 for SSL/TLS-encrypted connections.*
Connection Table	The Connection Table is a list of MQTT server IPv4 addresses, which can be used to define one or more failover MQTT server. This property is retained for use with legacy systems which used MQTT version 3.0 or 3.1, but typically this field should be left blank and instead use the URL_List table. See the **MQTT Option** property (below). NOTE that all IP addresses in the table use the same Connection Port. **Destination Address** – IP address of the MQTT server to connect to. If a connection fails to one address, the next address in the list is attempted, the MQTT process returns to the top of the list and attempts to walk through the list of servers in order (note, this behavior is different from the URL_List below). **Interface** – Device interface, which must match the Domain Name in the network configuration object, such as Ethernet, PPP, etc. (case-sensitive). This indicates which interface to use for the connection.
MQTT Option	Select the protocol version option to use for MQTT communication. Available options are: • ***Ver3.0 MQtt***. Use the 3.0 version of MQTT protocol, with the IP-based addresses defined in the **Connection Table** (above). This option is the default if this property does not exist (in older ACE objects) and does not use User Name and Password selection defined below. • ***Ver3.1 MQtt***. Use the 3.1 version of MQTT protocol, with the IP-based addresses defined in the **Connection Table**. All 3.1 or 3.1.1 MQTT options use the User Name/Password for authentication to the broker. • ***Ver3.1 MQtt with URLs***. Use the 3.1 version of MQTT protocol, using the list of server addresses defined in the **URL_List** (below). • ***Ver3.1.1 MQtt with URLs***. Use the 3.1.1 version of MQTT protocol, using the list of server addresses defined in the **URL_ List** (below).
User Name	User Name to use in MQTT protocol for authentication to the broker. This authentication is not performed when version 3.0 MQTT option is selected. *Enter a User Name up to 128 characters*.
Password	Password associated with the User Name for authentication to the broker. *Enter a Password up to 128 characters.* *If the **User Name** or **Password** field is set to **${MQTT_LOGIN}** and an entry is included in the GlobalText object with something like:* *Search Name=MQTT_LOGIN, Replacement Text=This_Is_A_Long_String_Up_To_512_Bytes* *then then field will be used from the GlobalText entry. This is useful if the MQTT system requires a user or password longer than 128 characters.*

URL_List	Click the **Edit Table** button to edit the list of MQ server addresses. This option is only used when the MQTT Option (above) is set to one of the choices "with URLs" – otherwise, the numeric **Connection Table** is used. At least one address is required in the **URL_List** table, and multiple addresses can be used for failover to multiple MQTT servers. When a connection is lost to one server, the next server in the list is attempted sequentially. **Broker FQDN** – Enter the IP address, URL or fully-qualified domain name (FQDN) for the network address of the MQTT server to connect to (up to 127 characters allowed). If a connection fails to one address, the next address in the list is attempted. All addresses in the table use the same Connection Port, and the port number should not be included in the FQDN table. Some examples of Broker FQDN are: • 10.73.1.20 • localhost • address.mybroker.com

When using MQTT Client with SSL/TLS encryption, you must configure the TLS Tunnels object to include one or more connections from the local host to a remote MQTT server (see TLS Tunnels*). To use more than one server address for backup connections, the STUNNEL Parameters should be set up with multiple 'localhost' addresses (127.0.0.x), and the MQTT Client object will be defined to connect to those local addresses, as shown in the following table:*

MQTT Client address list	*SSL/TLS 'stunnel' Parameters*	
(Connect Port=1883)	*Accept Connect*	*Connect To*
127.0.0.1	*127.0.0.1:1883*	address1.mybroker.com:8883
127.0.0.2	*127.0.0.2:1883*	address2.mybroker.com:8883
127.0.0.3	*127.0.0.3:1883*	address3.mybroker.com:8883

MQ_RBE_PR_Handler

The MQ_Rbe_Pr_Handler and MQ_Extra_Rbe_Pr_Handler are child objects under the MQClient or MQ_Extra_Clients objects. The MQ_RBE objects allow the Report by Exception (RBE) process to work through the MQ broker in a similar manner as RBE to the Elecsys HCP system. Whereas the Publish/Subscribe messages using the MQtt protocol typically require ISaGRAF logic to build topic strings and payloads, the MQ_RBE is designed to report unsolicited real-time data from an RTDB, defined solely using an ACE configuration, without requiring ISaGRAF program logic.

The RBE data packets are reported using a defined format, which can be processed through the message flows of an MQ broker as needed. Commands are also supported through the MQ_RBE process. The RediGate subscribes to the topics of "cmd" and "sys" in order to receive commands sent via the MQ-HCP or other publisher (the subscription for these topics is configurable; see Subscriptions). The "sys" topic may be published from a host system with a numeric payload to perform the following operations:

1. Restart the gateway
2. Send full update of all FieldUnit databases.
3. Send full update of this FieldUnit database.
4. Stop the Channel polling.
5. Start the Channel polling.
6. Health echo request
7. Walk the broker connection table to the next available IP

In order to use the MQ_RBE task, at least one Field Unit needs to have its "Produce RBEs" property set to "Yes". For all Field Units that have this setting, the RBE flag will be set for every point in the database that changes beyond its configured Deadband, and the MQ_RBE task will report that data to the MQ broker.

Note that the RBE data is only sent for a Field Unit that is marked as alive by the Master Channel. A unit will be marked as failed if any of its polls (with period less than the Scan Effective Limit) failed on the last attempt.

Attributes	Function
Object Type	MQ_Rbe_Pr_Handler, MQ_Extra_Rbe_Pr_Handler
Parent(s)	System Clients MQ_Client System Clients MQ_Extra_Clients
Instance	Must be 0.

Properties	Values
MQ RBE Pacing	Milliseconds to wait between sending each RBE packet. *T* *he RBE task periodically checks the RBE flags set for each point in the RTDB to see if data has changed. If so, all changed points are sent to the broker. This parameter defines how often to perform this check.*
Console_ID	Text string which may be inserted into topic strings for RBE messages, to identify the destination MQ-HCP when using multiple HCP's. *The **Console_ID** text, if configured, is substituted into the 'sys', 'cmd', and 'file' Control Topics in place of the "Console_ID" (see Subscriptions object). If the **Console_ID** field is left empty, the field is omitted from the subscribe topic string.*
Subscribe Topic Rules	Select the topic publish rule for MQ RBE messages, which are also the rules that will be used by other clients who subscribe to the broker for the RBE data (such as the MQ_HCP). *When publishing RBE messages, the topic always begins with the topic string "RBE", and the **Console_ID** field configured in this MQ_RBE object.. The topic string can also include the following data specific to this unit configuration:* • *"UnitName" or "UnitAddress" (taken from the top level System object in ACE)* • *"RTUName" or "RTUAddress" (taken from the Field Unit object)* • *and optionally the "ChannelName" or "ChannelNumber" (taken from the Master Channel object). The Channel information can be excluded if all RTUs in this unit have unique names or addresses.* Select the RBE publish **Subscribe Topic Rule** from the following options, where HcpId, UnitName, UnitAddress, ChannelName, ChannelNumber, RtuName, and/or RtuAddress in the topic would contain the actual property data in the configuration: • RBE/*HcpId/UnitName/ChannelName/RtuName* • RBE/*HcpId/UnitName/RtuName* • RBE/*HcpId/UnitAddress/ChannelNumber/RtuAddress* • RBE/*HcpId/UnitAddress/RtuAddress*
DBM RBE Cl assification	This option is used in systems using redundant MQTT servers. In this case, the MQTT RBE/PR objects in the ACE configuration will use an instance number (0=primary, 1=secondary). If only one MQTT connection is used, the MQ_Client/MQ_Rbe_Pr_Handler process may be configured to use the other RBE instance: *Select 'RBE 0 DBM Flag' to use the primary (RBE0) process for MQ_RBE.* *Select 'RBE 1 DBM Flag' to use the secondary (RBE1) process for MQ_RBE. In this case, standard HCP RBE/PR connections would be configured to use instance 0.* If both primary and secondary HCP connections are used in a configuration, the MQ_Extra_Clients/MQ_Extra_Rbe_Pr_Handler process may be configured to use the alternate RBE instances: *Select 'TERTIARY RBE process for DBM' for the third (RBE2) process for MQ_RBE.* *Select 'QUATERNARY RBE process for DBM' for 4th (RBE3) process for MQ_RBE.*
Enable RBE List	This list allows the MQ_RBE process to filter which Channel/Field Unit configurations will report their data using the automatic RBE process. If this table is left blank (or is omitted in older versions of the configuration), then all configured RTDB database fields will report data on all MQTT clients. If one or more rows are entered in the table, then only the configured FieldUnits will be reported through this MQTT Client. Click the **Edit Table** button to edit the list of Field Units. **Channel** – Enter the channel number (0-16) of the Field Unit to include in RBE reporting. **RTU to Enable** – Enter the Field Unit address of the device to include in RBE reporting.

MQ-RBE Subscriptions

The Subscriptions object is an optional child object under MQ_Rbe_Pr_Handler and allows control of several aspects of the MQTT connection. If the Subscriptions object is omitted from a configuration, these parameters are set to default values of subscription topics, host synchronization, and publish options.

Attributes	Function
Object Type	Subscriptions, Extra_Subscriptions
Parent(s)	System Clients MQ_Client System Clients MQ_Extra_Clients
Instance	Must be 0.

Properties	Values
Control 'sys' topic	Select whether to subscribe to the MQTT 'sys' topic. The 'sys' topic subscription is used to allow a remote host to send system level commands (see MQ_RBE_PR_Handler). • ***sys/Gateway/#*** *– issue subscription where "Gateway" is replaced by the Unit Name in the top-level System object (default, if Subscriptions object is omitted)* • ***sys/HCP_ID/Gateway/#*** *– issue subscription where "Gateway" is replaced by the Unit Name, and HCP_ID is taken from the parent MQ_RBE object.* • ***Disable subscription to 'sys' topic*** *– this option prevents 'sys' topics from being received on this device from a host. Note that this setting also prevents Health Echo commands, so an appropriate (non-Health Echo) option must be chosen for Host Synchronization.*
Control 'cmd' topic	Select whether to subscribe to the MQTT 'cmd' topic. The 'cmd' topic subscription is used to allow a remote host to set or control data values in the FieldUnits on this device. This could be used in a situation requiring a field device to be monitored but commands must be explicitly prevented. *cmd/Gateway/# – issue subscription where "Gateway" is replaced by the Unit Name in the top-level System object (default, if Subscriptions object is omitted)* *cmd/HCP_ID/Gateway/# – issue subscription where "Gateway is replaced by the Unit Name, and HCP_ID is taken from the parent MQ_RBE object.* *Disable subscription to 'cmd' topic – this option prevents 'cmd' topics from being received on this device from a host.*
Control 'file' topic	Select whether to subscribe to the MQTT 'file' topic. The 'file' topic subscription is used to allow a remote host to publish a file to this device via MQTT. This could be used with some additional logic to process configurations or other files. *file/Gateway/# – issue subscription where "Gateway" is replaced by the Unit Name in the top-level System object (default, if Subscriptions object is omitted)* *file/HCP_ID/Gateway/# – issue subscription where "Gateway is replaced by the Unit Name, and HCP_ID is taken from the parent MQ_RBE object.* *Disable subscription to 'file' topic – this option prevents 'file' topics from being received on this device from a host.*

Host Synchronization	Select how (or whether) to synchronize MQTT communication with a host system. Synchronization is typically used when this device is publishing through multiple MQTT servers, and one or more hosts is also connected to the MQTT servers. This ensures that the device and the host system are communicating with each other through the same MQTT server, so that commands and data are sent and received directly. *Require Health Echo from host – Host must periodically send out a 'sys' command with Health Echo and this device will respond with an echo, ensuring synchronized communication (default, if Subscriptions object is omitted). If the Health Echo is not received within two times the Keep Alive period configured in MQClient, the MQClient will disconnect from the current MQTT server and try the next address.* *Subscribe to 'STATE/HostName' – (see Host Name option below) This synchronization method reduces network traffic by not requiring constant application-level Health Echo messages.* *In this case, the host should publish a message (with Retain flag set) with topic of "STATE/HostName" (where "HostName" is configured to identify the host). The payload of the message should be "CONNECTED" (all caps). It should also lodge a Last Will and Testament with the server using the same topic and a payload of "DISCONNECTED."* *When a RediGate device connects to the server and subscribes to 'STATE/HostName', the server will deliver the retained message immediately with the payload "CONNECTED", thus informing the device that the host application is currently connect on the same server.* *If this STATE message is not received within two times the Keep Alive period, the MQ Client will move to the next server. If the server disconnects from the server, the RediGate will receive the "DISCONNECTED" message and will immediately move on to the next server, attempting to locate a server to which the host is simultaneously connected.* *No Host Synchronization – If only a single MQTT server is used, this option can be used. The MQ Client will not look for a Health Echo or STATE message but will remain connected to the server. Synchronization with a host is ensured as long as the host is also connected to the server.*
RBE Data at Restart	Select how to publish data on startup or upon reconnection to an MQTT server. *Send Only Non-Zero data registers – Upon connection, publish only non-zero values in RTDB registers (default, if Subscriptions object is omitted). Upon receiving the Birth Certificate from this device, the host must zero its entire database of values, so that non-zero data received from the device will ensure host and device have identical data. This option reduces the amount of published traffic upon connection.* *Send All data including Zero/Null registers – Upon connection, all RTDB registers (except those configured as "Non-RBE") will be published even if they contain zeroes. This causes more data traffic on reconnection, but it ensures the host is informed of all available points, values, and identifications (tags or register numbers) at the time of connection.*
Publish Tag Names	Select whether to publish tag names associated with RTDB register locations. Publishing registers with tag names requires the configuration of the optional Tag Names object under each RTDB (see Tag Names). *Don't Publish Tag Names – Upon connection, publish only RTDB registers by number (default, if Subscriptions object is omitted). Upon receiving the Birth Certificate from this device, the host must zero its entire database of values, so that non-zero data received from the device will ensure host and device are in synch.* Publish Tag Names on MQTT Connect – Upon connection, publish a list of the configured Tag Names associated with RTDB register addresses for the Field Unit. Subsequently, all data will be published using the RTDB register addresses, but the host system can use the initial list to correlate addresses with names for display purposes. Publish GZIP'd Tag Names on MQTT Connect – Same option as previous, but the initial publication of Tag Names and register addresses is sent in the MQTT payload in zipped format using 'gzip' compression. This reduces the bandwidth and packet size when publishing a large database of tag names.
Host Name	Enter the Host Name to use in the 'STATE/HostName' subscription, if using that option for Host Synchronization *Host Name must be a maximum of 80 characters. If using the State/HostName option and this field is left blank, the Host Name in the subscription will default to the HCP_ID in the MQ_RBE object.*

Sparkplug (SparkplugB_RBE)

The Sparkplug B object provides a connection to an MQTT broker that expects data to be published in the Sparkplug B format, and open source MQTT payload developed and maintained by Cirrus-Link Solutions. More information about the Sparkplug B payload can be found here: http://www.cirrus-link.com/oem-device-data-integration/.

NOTE that Sparkplug does not support sending UINT32 or UINT64 registers. The RTDB of Field Units sending data to Sparkplug should use SINT32 or SINT64 data types instead.

Attributes	Function
Object Type	SparkPlugB_RBE
Parent(s)	System Clients
Instance	Must be 0.

Properties	Values
RBE Flag	Select which of four RBE flags to use for checking changed data (make sure not to use the same as any other process that might use the same RBE flag, such as MQTT, HCP, DNP).
RBE Pacing	Enter the number of milliseconds to wait between RBE messages
Accept NODE Cmd Topics	Select whether to subscribe for NODE (Gateway) command topics.
Accept DEVICE Cmd Topics	Select whether to subscribe for DEVICE command topics (output commands to protocol)
Accept File Topics	Select whether to subscribe for file topics
Group Name	GroupName is Appended to 'spBv1.0' namespace topics (unless property is left blank), used to filter published data in groups.
STATE Topic	The STATE topic is used to ensure an active end-to-end connection to Host, or switch to next MQTT server
RBE Processing	Only process RBE RTDB Registers with TagNames?
Account	Account name for MQTT Access Control List (255 chars)
Password	Password for MQTT Access Control List (255 chars)
MQTT Port	MQTT Broker Connection Port (default is 1883 for unencrypted, or 8883 for TLS)
MQTT Keep Alive	Enter the interval (in Seconds) to send MQTT keep alive. Expect response within 1.5 times Keep Alive, or disconnect and try the next server.
MQTT Connect Delay	Enter the time (in Seconds) to delay after loss of connection before reconnect.
Enable RBE List	Enter list of channel/unit to publish. If table is empty, publish ALL RTUs with RBE data enabled.
URL_List	Enter a list of URLs or FQDN for MQTT server failover.

Store and Forward

The Store and Forward object allows real-time changes in the RTDB data to be stored into a CSV format file on the RediGate's removable SD memory card. The data stored may then be delivered as historical values in a variety of ways. Then configuration options for Store & Forward are described below.

Attributes	Function
Object Type	Store and Forward
Parent(s)	System Clients
Instance	Must be 0.

Properties	Values

Operation Mode	Select when to store RTDB values as historical records. • **Store changes only on Link Failure** – Only store values when the "Process to Monitor" is in a disconnected state. The values stored will be marked as "Unpublished." • **Always store changes** – Store all RBE values from the RTDB into the CSV file. If the "Process to Monitor" is in a connected state, the data will be marked as "Published." If the "Process to Monitor is in a disconnected state, the data will be marked as "Unpublished."
Process to Monitor	Select which Process (or a TCP Port) should be monitored to determine the "Link Failure" condition. If the Operation Mode is set to store changes only on Link Failure, values will be stored as Unpublished data when the link is broken. This property also determines when to deliver data automatically after the link is restored – when the link is restored, the unpublished historical values will be sent using the Delivery Technique, below. The following options apply to Process to Monitor: • **None** – Don't monitor any process for link status (requires the **Always store changes** option for the Operation Mode). • **MQClient** – Monitor the MQClient process for connection to a broker, and store as Unpublished if not connected. • **MQttExtra instance 0** (or 1) – Monitor the MQTT Extra process with instance number 0 or 1 for connection to a broker, and store as Unpublished if not connected. • **Sparkplug-B instance 0** – Monitor the Sparkplug-B process with instance number 0 for connection to a broker, and store as Unpublished if not connected. • **Monitor a TCP Port link status** – Monitor any TCP port connection for being in an "Established" state, and if not connected then data will be stored as Unpublished. The TCP "Port to Monitor" is configured in the following property. (Note: in this and several of the subsequent options, "published" is used loosely to imply some form of real-time data delivery and does not necessarily imply delivery via MQTT publishing.) • **Modbus Net Slave instance 0** (or 1) – Monitor Modbus NetSlave process with instance number 0 or 1 for connection to a Modbus host, and store as Unpublished if not connected. • **Hcp Rbe/Pr instance 0** (or 1) – Monitor connection to Elecsys HCP with instance number 0 or 1, and store as Unpublished if not connected. • **Dnp3 Slave instance 0** (or 1) – Monitor NetDnpSlave process with instance number 0 or 1 for connection to a DNP3 TCP host, and store as Unpublished if not connected.
Port to Monitor	TCP Port to monitor – this option only applies if using the **Monitor a TCP Port link status** option above.
RBE Flag	Select which of four RBE flags (0 to 3) to use for checking changed data. **Important**: make sure not to use the same RBE Flag used by any other process for RBE processing, such as MQTT, HCP, or DNP3.
Pacing	Minimum interval delay between checking the RBE Flag for changed data in order to store values to the archive. • If the Pacing is set to 1 or greater, the CSV file will contain "EpochTime" in seconds for the time of the event. • If the Pacing is set to 0, the effective storage rate is limited to something around a quarter of a second, and the CSV file will contain "EpochTimeMS" in milliseconds for the time of the event.
RBE Processing	Only process RBE RTDB Registers with TagNames? • Select **Yes** to store only RTDB registers with RBE data types and with a configured Tag name. • Select **No** to store all RTDB registers with RBE data types.
External Storage	Use which external Memory to store historical data (currently only supports SD card). Values are stored to CSV files in /tmp/sdcard1/SNF***aa_bbbbb***/, where ***aa*** is the Channel number instance, and *bbbbb* is the RTU address. Files are named CHAN***aa***~RTU-***b***~***yyyymmdd***.csv, where ***aa*** is the Channel number instance, ***b*** is the RTU address, and ***yyyy mmdd*** is the year/month/day of the date when the values were stored. Such as: /tmp/sdcard1/SNF05_00009/CHAN05~RTU-9~20180731.csv See the RediGate Diagnostics Manual for a menu option to view the contents of Store & Forward CSV files.
Files to Retain	Number of recent daily files to retain stored historical data before purging. Data is stored in one CSV file per day for each device.

Delivery Technique	Select how to deliver historical data from the CSV files when link is restored. When publishing individual historical values, only data marked Unpublished are sent. • **No automatic delivery, user will retrieve** – Select this value if you plan on manually retrieving the data from the SD card using a technique such as an FTP upload. • **Send Unpublished values via JSON-Rbe_0** (or 1) – Data will be published using instance 0 or 1 of the MQTT_Extra_Clients3_1 object. This requires a JSON MQTT process to be configured (does not work with MQ-RBE using the MQ_Extra_Rbe_Pr_Handler object). • **Send Unpublished values via Sparkplug-B_0** – Data will be published using the Sparkplug B connection in the "Historical" metric. • **Run Reconnect Script After Link Reconnect** – On reconnection to the host process selected in Process to Monitor, the RediGate will run the script located in the file location defined in the "Reconnect Script" property
Reconnect Script	BASH Script or command line to run after reconnection – this option only applies if using the Delivery Technique of "Run Reconnect Script After Link Reconnect." This option might be used, for example, if a script is included which performs an FTP operation to upload the latest .csv files somewhere and then deletes the current files.
QOS	Quality of Service for delivery of historical data (applies to JSON or Sparkplug delivery technique). Currently, the only option is QOS-0.
Send Data Format	Unused option – reserved for future use.
Enable RTU List	Enter list of Master Channel/Field Unit RTDBs to use with Store & Forward. If table is empty, store and publish ALL Field Units with RBE data enabled.

Store & Forward Payload Definitions

When the "Delivery Technique" property is set to JSON or Sparkplug B, the RediGate will publish the stored .CSV in either a JSON or Sparkplug payload. In this section, we define the payload that is published on reconnect for each of the two options:

JSON-RBE

Elecsys created the JSON-RBE payload specification for easy integration into 3rd party applications. Historical data is plublished in a text-based JSON object that can be easily parsed. More information about the historical JSON-RBE payload can be found here: JSON-RBE MQTT Payload Format#RBEMQTTPayloadFormat-HistoricalDataPayload.

Sparkplug B

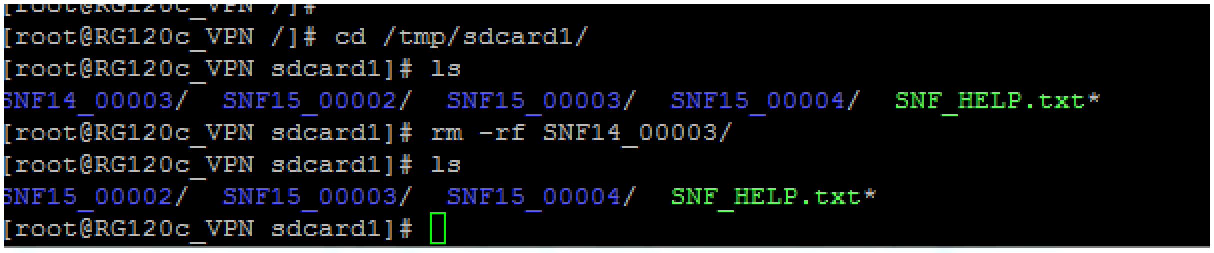
The Sparkplug B specification is an efficient, binary MQTT payload definition created by Cirrus-Link solutions and supported by SCADA applications such as Ignition Automation's Ignition platform. See the section on Sparkplug B configuration. Information about the Sparkplug B specification and the historical payload can be found on Cirrus-Link's website: http://www.cirrus-link.com/oem-device-data-integration/.

Troubleshooting

See RediGate Diagnostics Manual (Directory Services menu 21, Store-N-Forward File) for information regarding how to diagnose issues with the store and forward object. You can view diagnostic information about the store and forward function by selecting the "STORFWD_ Status" from the "Task Diagnostics" menu (see RediGate Diagnostics Manual, Diagnostic Services option 6).

To edit or remove the stored CSV data files:

1. Login to the RediGate's MMI using the "root" user credentials (email idc-support@elecsyscorp.com for default root password)
2. Run the command "cd /tmp/sdcard1"
3. Run the command "ls" to show the folder contents
4. Delete each folder within the sdcard1 directory by running the command: `rm -rf` ***`foldername`***



```
[root@RG120c_VPN /]# cd /tmp/sdcard1/
[root@RG120c_VPN sdcard1]# ls
SNF14_00003/  SNF15_00002/  SNF15_00003/  SNF15_00004/  SNF_HELP.txt*
[root@RG120c_VPN sdcard1]# rm -rf SNF14_00003/
[root@RG120c_VPN sdcard1]# ls
SNF15_00002/  SNF15_00003/  SNF15_00004/  SNF_HELP.txt*
[root@RG120c_VPN sdcard1]# 
```

NTP Client

The NTP Client Sync function allows the clock to be synchronized with one or more Network Time Protocol (NTP) servers.

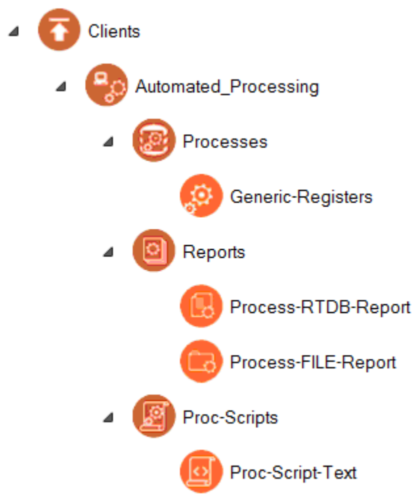
Attributes	Function
Object Type	NTP Client
Parent(s)	System Clients
Instance	Must be 0.

Properties	Values
NTP Server #1-6	Enter one or more IP address of the NTP servers to which to connect. NTP uses algorithms to average the errors between different time servers, and to detect servers which are not reliable. *If any server address is set to 0.0.0.0, all subsequent addresses in the list are ignored.* A couple of publically available NTP servers you can try are 129.6.15.28 and 129.6.15.29

Upon startup, the NTP synchronization is called with a parameter requesting an immediate clock update. This will allow an erroneous clock setting to be more quickly adjusted. Subsequent updates may produce a gradual shift in the clock until the time is precisely correct, based on the normal NTP correction algorithm. The RediGate also schedules the NTP operation to be restarted every 24 hours with an immediate clock update.

Automated Processing

The Automated Processing section of the RediGate configuration is a collection of processes designed primarily to collect data from various sources (RTDB, text files; representing different flow computers and devices) and format them into a "report" which may be published via MQTT, HTTP, and/or stored to the gateway for later retrieval.



See the section Automated Processing for documentation on this feature.

Terminal Client (TermClient)

The Terminal Client process connects to a TCP/IP server on a network, and bridges that network connection with a local serial port. This is distinguished from the Terminal Server by the fact that the RediGate initiates the connection to the TCP/IP server, as opposed to an external network client making the network socket connection to the RediGate. The Terminal Client process is similar to the "Reverse Telnet" option available in many routers.

The Terminal Client IP connection may be initiated always at startup of the system or upon receiving any data from a device on the serial port, and it also provides an option as a "modem emulator" whereby the connected serial device can use AT commands to direct which IP address should be connected to.

Attributes	Function
Object Type	TermClient
Parent(s)	System Clients
Instance	Use a unique Instance number for each Terminal Client object. The Terminal Client object must have at least one child Host Connection object defined.

Properties	Values
Serial Port	From the dropdown list, select the serial port which should be used for connection to this Terminal Client TCP network port. *Sele ct the Serial Port from among standard serial ports (COMx) or one instance of a pair of Virtual Port pair (VirCOM **xx**a or **xx**b) if connecting this Terminal Client port to another process using Virtual Serial Ports. Make sure the Async port is defined under Networks.*
Serial Buff Size	Maximum number of bytes which will be put into an IP packet to be sent to the network server. Once receiving this quantity of bytes on the serial port, a TCP packet will be sent immediately. *The actual amount of data bytes may be less than the Serial Buff Size, if the Demark Timer (below) times out before the Serial Buffer is full.*
Demark Timer	Maximum time (in milliseconds) to wait before creating and sending a packet regardless of how many data bytes were received on the serial port. *This indicates the assumed "demarcation" time between serial packets. If a serial device is periodically sending messages at a defined interval, setting this Demark Timer too high could cause multiple serial packets to be clumped together in one TCP packet, which may not be desirable. Setting this value too low might cause half of a packet to be sent prematurely if there is a momentary glitch or pause in the serial data stream.*
Client Reconnect Delay	Enter the time (in seconds) that the Terminal Client will wait after a failure to connect, before attempting to reestablish connection with the Server. *This only applies when the connection is unable to be established. If the connection is made and then lost, reconnection will be attempted immediately.*

Ok AT Commands	Select whether to echo an "OK" to AT commands entered at the Terminal Client serial port. The OK is similar to communicating with a modem over its serial port.
DTR Indicates O nline	Select whether DTR indicates IP connection state. *If set to 'Yes', the serial port's DTR signal will be asserted to a positive voltage when the IP connection is established with a remote server, and will be de-asserted when the IP connection is lost. This emulates the Data Terminal Ready functionality of a dial-up modem, giving a physical indication that an active connection is present.*
Mode Flag	Select the connection mode of the Terminal Server. ***Always*** – Select this option to connect automatically upon system restart or upon the IP socket connection being terminated. If using the "Always" mode of connection, there must be only one child Host Connection object with its Dial String set to an empty field. The Terminal Client will use the IP address of that Host Connection object to automatically connect to the end device. ***Any Data*** – Select this option to connect the Terminal Client to the remote server only when data is received on the serial port. If using the "Any Data" mode of connection, there must be only one child Host Connection object with its Dial String set to an empty field. The Terminal Client will use the IP address of that Host Connection object to connect when data is received on the serial port. ***ATDT*** – Select this option to connect the Terminal Client only if an "ATDT###" message is received on the serial port, where ### is some alphanumeric string. When using the “ATDT” option, there may be many child Host Connection objects defined under this Terminal Client object. The Host Connection entries should have their Dial String configurations set to unique ATDT### values. The “ATDT” option causes the Terminal Client to act as a modem emulator. The connected serial device acts as if it were connecting using a dial-up modem, where each ATDT dial sequence tells the Terminal Client to connect to a destination IP server, rather than dialing over a PSTN telephone network. ***DCD*** – Select this option to connect the Terminal Client only when the DCD signal on the RS-232 serial port is raised to a positive voltage. This option allows a physical voltage input on the serial port to trigger the network socket connection. If using the “DCD” mode of connection, there should be only one child Host Connection object with its Dial String set to an empty field. The Terminal Client will use the IP address of that Host Connection object to connect when the DCD control signal is received. ***ATDT or DCD*** – Select this option to connect the Terminal Client in one of two modes described above (the “ATDT” or “DCD” modes). When choosing this option, it is required that one or more Host Connection objects be defined with a configured Dial String, and there should also be only one Host Connection object defined with its Dial String set to an empty field. If an “ATDT###” message is received on the serial port, the matching Host Connection object is used, and the Terminal Client connects to that IP address. If an active DCD signal is present, the Terminal Client connects to the IP address defined in the first Host Connection object containing an empty Dial String field.
Time to Live	Number of seconds to close socket after inactivity (0 disables TTL)

Terminal Client Host Connection (HostCon)

The Host Connection objects are used in conjunction with the Terminal Client to configure one or more IP addresses to which the Terminal Client will connect, and to allow the Terminal Client to be used as a modem emulator.

Attributes	Function
Object Type	HostCon
Parent(s)	System Clients TermClient
Instance	Use a unique Instance number for each Host Connection object.

Properties	Values

Dial String	Enter the Dial String, if applicable, in the form “ATDT###”, where ### is some unique alphanumeric string (up to 30 characters). This field may be left blank. *If using the Terminal Client in “ATDT” or “ATDT or DCD” connection modes, enter the ATDT#### string that is used to make a connection. Each ATDT string should be configured should be unique among all Host Connection objects under a given Terminal Client.* *The Dial String has an alternate function when using the Terminal Client to connect to a named server (URL, FQDN). If the Connection Table IP address is set to 0.0.0.0 and Interface set to "DNS", then you can enter a URL in the Dial String field.* *If using any other connection mode, leave this field blank. In those cases, the IP address(es) defined in the Connection Table become the default IP address used by the Terminal Client to which connection is made.*
Connect Msg	Enter a string to echo back to the serial device when the Terminal Client is connected to the destination IP address. This is designed to emulate the "Connect" message given by a dial-up modem to a serial terminal program when dialing a PSTN network.
Fail Msg	Enter a string to echo back to the host if the connection cannot be established. This is designed to emulate the "Fail" message given by a dial-up modem to a serial terminal program when failing a connection to a PSTN device.
Disconnect On	Select from the dropdown menu the conditions under which the Terminal Client should disconnect from the IP network server: ***Never*** – Never terminate the Terminal Client from the remote server (remote side can drop the connection at any time). **+++** - Disconnect Terminal Client connection from the remote server if the string "+++" is received on the serial port. This must be exactly three pluses in a row. This function emulates the modem attention string often used prior to hanging up a PSTN dial connection. ***DCD Drop*** – Disconnect the Terminal Client from the remote server when the DCD input on the RS-232 serial port goes to a low (inactive) state. This can be used in conjunction with the DCD connect option in the Terminal Client, to allow the DCD signal to both connect and disconnect; however, the two configuration options are independent and don't have to be used together. ***+++ or DCD Drop*** – Disconnect the Terminal Client from the remote server either upon receiving the string "+++" on the serial port, or when an inactive DCD signal is detected on the serial port.
Echo Commands	Select whether to echo all commands. The options are: ***No*** – Do not echo characters typed from the serial port. ***To Async*** – Echo characters to serial port only. ***To Async & IP*** – Echo characters received on the serial port to both the serial port and the remote connected IP server.
Fail Over Time	Enter the time (in seconds) to wait after a failed connection attempt until attempting to connect to the next Destination Address, defined in the Connection Table below.
Remote IP Port	Port number for the Terminal Client to connect to. If the Host Connect object has a list of Destination IP addresses in the Connection Table, it will always use the same port number with each IP address.
Connection Table	Click the **Edit Table** button to define the IP address(es) and network interface for the Host Connection object. **Destination Address** – IP address for the Terminal Client to connect to. Configuring a list of more than one IP address provides a fail-over connection in case one connection is unable to connect. **Interface** – Device interface, which must match the Domain Name in the ACE network configuration objects, such as Ethernet, PPP, etc. (case-sensitive). This tells which interface to use for the connection. To use a named server as the destination (URL or FQDN up to 30 characters) in the Terminal Client instead of an IP address, the following things must be configured: • Destination Address must be **0.0.0.0** • Interface must be **DNS** • Dial String must be the URL address of the connection The DNS feature currently only supports a single URL destination (not multiple failover addresses). Also make sure under Networks to configure the "DNS Servers" object or enable the "Use Peer DNS" option in the CellModem object.

Global Texts

The Global Text Function allow users to create variables that can be used in the MQTT Clients objects.

Attributes	Function
Object Type	Global Text
Parent(s)	System Clients
Instance	Must be 0.

Properties	Values
Extra Text	Click the **Edit Table** button to define the variables and corresponding values that will be usable in other areas of the configuration **Search Name**: Variable name for string replacement. This value will be accessible within an MQTT Client object by typing the name ${Search Name} **Replacement Text:** Up to 512 chars to replace the original ${Search Name}

There are a number of pre-defined global text objects that are built-in to the RediGate OS, and thus cannot also be defined in the "Extra Text" field. These pre-defined values are:

Variable Name	Function
${GATEWAY}	Returns the System Unit Name of the current configuration
${SERIAL}	Returns the serial number of the RediGate
${UUID}	Returns a universally unique UUID number, commonly used in the MQTT_Extra_Clients3_1 Client ID property